INVESTOR IN PEOPLE

The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

PRIORITY DOCUMENT

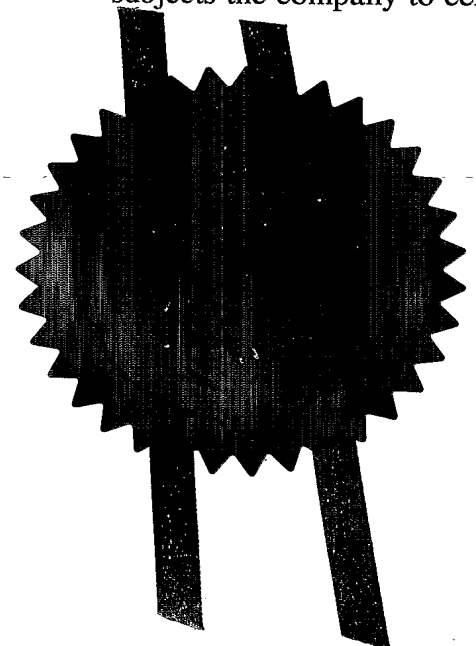
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.



Signed

Dated 3 December 2004

Patents Form 1/77

Patents Act 1977
(Rule 16)

The Patent Office

Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

The Patent Office

Cardiff Road
Newport
Gwent NP9 1RH

1. Your reference — IP/P7345/1

2. Patent application number *(The Patent Office will fill in this part)* — 0327627.6 — 28 NOV 2003

3. Full name, address and postcode of the or of each applicant *(underline all surnames)*

QINETIQ LIMITED

Registered Office 85 Buckingham Gate
London SW1E 6PD
United Kingdom

Patents ADP number *(if you know it)* 8307431001

If the applicant is a corporate body, give the country/state of its incorporation — GB

4. Title of the invention

ANOMALY DETECTION

5. Name of your agent *(if you have one)* — Arthur Wyn Spencer WILLIAMS

"Address for service" in the United Kingdom to which all correspondence should be sent *(including the postcode)*

QINETIQ LIMITED
IP Formalities
A4 Bldg
Cody Technology Park
Ively Road
Farnborough
Hants GU14 0LX United Kingdom

Patents ADP number *(if you know it)* 8242497001

6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and *(if you know it)* the or each application number	Country	Priority application number *(if you know it)*	Date of filing *(day / month / year)*

7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application	Number or earlier application	Date of filing *(day / month / year)*

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:*
   a) *any applicant named in part 3 is not an inventor, or*
   b) *there is an inventor who is not named as an applicant, or*
   c) *any named applicant is a corporate body. See note (d))*

Yes (b)

Patents Form 1/77

9. Enter the number of sheets for any of the following items you are filing with this form. Do not count copies of the same document

Continuation sheets of this form	0
Description	20
Claim(s)	4
Abstract	1
Drawing(s)	1 only DM

10. If you are also filing any of the following, state how many against each item.

Priority documents	0
Translations of priority documents	0
Statement of inventorship and right to grant of a patent *(Patents Form 7/77)*	4
Request for preliminary examination and search *(Patents Form 9/77)*	1
Request for substantive examination *(Patents Form 10/77)*	0
Any other documents *(please specify)*	0

11. I / We request the grant of a patent on the basis of this application.

Signature AWS Williams Date 28/11/2003

12. Name and daytime telephone number of person to contact in the United Kingdom: Mrs Linda Brockshaw 01252 392722

Warning

After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under Section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you live in the United Kingdom, Section 23 of the Patents Act 1977 stops you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent of the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.

Notes

a) If you need help to fill in this form or have any questions, please contact the Patent Office on 0645 500505.

b) Write your answers in capital letters using black ink or you may type them.

c) If there is not enough space for all the relevant details on any part of this form, please continue on a separate sheet of paper and write "see continuation sheet" in the relevant part(s). Any continuation sheet should be attached to this form.

d) If you have attached 'Yes' Patents Form 7/77 will need to be filed.

e) Once you have filled in the form you must remember to sign and date it.

f) For details of the fee and ways to pay please contact the Patent Office.

Patents Form 1/77

Anomaly Detection

This invention relates to anomaly detection, and to a method, an apparatus and computer software for implementing it. More particularly, although not exclusively, it relates to detection of fraud in areas such as telecommunications and retail sales by searching for anomalies in data.

It is known to detect fraud with the aid of fraud management systems which use hand-crafted rules to characterise fraudulent behaviour. The rules are generated by human experts in fraud, who supply and update them for use in fraud management systems in detecting fraud. The need for human experts to generate rules is undesirable because it is onerous, particularly if the number of possible rules is large or changing at a significant rate.

It is also known to avoid the need for human experts to generate rules; i.e. artificial neural networks are known which learn to characterise fraud automatically by processing training data. They then detect fraud indicated in other data from characteristics so learned. However, neural networks characterise fraud in a way that is not immediately visible to a user and does not readily translate into recognisable rules. It is important to be able to characterise fraud in terms of breaking of acceptable rules, so this aspect of neural networks is a disadvantage.

Known rule-based fraud management systems can detect well-known types of fraud because experts know how to construct appropriate rules. In particular, fraud over circuit-switching networks is well understood and can be dealt with in this way. However, telecommunications technology has changed in recent years with circuit-switching networks being replaced by Internet protocol packet-switching networks, which can transmit voice and Internet protocol data over telecommunications systems. Fraud associated with Internet protocol packet-switching networks is more complex than that associated with circuit-switching networks: this is because in the Internet case, fraud can manifest itself at a number of points on a network, and experts are still learning about the potential for new types of fraud. Characterising complex types of fraud manually from huge volumes of data is a major task. As telecommunications traffic across packet-switching networks increases, it becomes progressively more difficult to characterise and detect fraud.

The present invention provides a method of anomaly detection characterised in that it incorporates the steps of:-

a) developing a rule set of at least one anomaly characterisation rule from a training data set and any available relevant background knowledge, a rule covering a proportion of positive anomaly examples of data in the training data set, and

b) applying the rule set to test data for anomaly detection therein.

in an alternative aspect the present invention provides an automated method of anomaly detection characterised in that it comprises using computer apparatus to execute the steps of:-

a) developing a rule set of at least one anomaly characterisation rule from a training data set and any available relevant background knowledge, a rule covering a proportion of positive anomaly examples of data in the training data set, and

b) applying the rule set to test data for anomaly detection therein.

The method of the invention provides the advantage that it obtains rules from data, not human experts, it does so automatically, and the rules are not invisible to a user.

Data samples in the training data set may have characters indicating whether or not they are associated with anomalies. The invention may be a method of detecting telecommunications or retail fraud from anomalous data and may employ inductive logic programming to develop the rule set.

Each rule may have a form that an anomaly is detected or otherwise by application of the rule according to whether or not a condition set of at least one condition associated with the rule is fulfilled. A rule may be developed by refining a most general rule by at least one of:

a) addition of a new condition to the condition set;

b) unification of different variables to become constants or structured terms.

A variable in a rule which is defined as being in constant mode and is numerical is at least partly evaluated by providing a range of values for the variable, estimating an accuracy for each value and selecting a value having optimum accuracy. The range of values may be a first range with values which are relatively widely spaced, a single optimum accuracy value being obtained for the variable, and the method including selecting a second and relatively narrowly spaced range of values in the optimum

accuracy value's vicinity, estimating an accuracy for each value in the second range and selecting a value in the second range having optimum accuracy.

The method may include filtering to remove duplicates of rules and equivalents of rules, i.e. rules having like but differently ordered conditions compared to another rule, and rules which have conditions which are symmetric compared to those of another rule. It may include filtering to remove unnecessary 'less than or equal to' ("lteq") conditions. Unnecessary "lteq" conditions may be associated with at least one of ends of intervals, multiple lteq predicates and equality condition and lteq duplication.

The method may include implementing an encoding length restriction to avoid overfitting noisy data by rejecting a rule refinement if the refinement encoding cost in number of bits exceeds a cost of encoding the positive examples covered by the refinement.

Rule construction may stop if at least one of three stopping criteria is fulfilled as follows:

a) the number of conditions in any rule in a beam of rules being processed is greater than or equal to a prearranged maximum rule length,

b) no negative examples are covered by a most significant rule, which is a rule that:

   i) is present in a beam currently being or having been processed,

   ii) is significant,

   iii) has obtained a highest likelihood ratio statistic value found so far, and

   iv) has obtained an accuracy value greater than a most general rule accuracy value, and

c) no refinements were produced which were eligible to enter the beam currently being processed in a most recent refinement processing step (32).

A most significant rule may be added to a list of derived rules and positive examples covered by the most significant rule may be removed from the training data set.

The method may include:

a) selecting rules which have not met rule construction stopping criteria,

b) selecting a subset of refinements of the selected rules associated with accuracy estimate scores higher than those of other refinements of the selected rules, and

c) iterating a rule refinement, filtering and evaluation procedure (32 to 38) to identify any refined rule usable to test data.

In another aspect, the present invention provides computer apparatus for anomaly detection characterised in that it is programmed to execute the steps of:-

a) developing a rule set of at least one anomaly characterisation rule from a training data set and any available relevant background knowledge, a rule covering a proportion of positive anomaly examples of data in the training data set, and

b) applying the rule set to test data for anomaly detection therein.

In a further aspect, the present invention provides computer software for use in anomaly detection characterised in that it incorporates instructions for controlling computer apparatus to execute the steps of:-

a) developing a rule set of at least one anomaly characterisation rule from a training data set and any available relevant background knowledge, a rule covering a proportion of positive anomaly examples of data in the training data set, and

b) applying the rule set to test data for anomaly detection therein.

The computer apparatus and computer software aspects of the invention may have preferred features equivalent *mutatis mutandis* to those of the method aspect.

In order that the invention might be more fully understood, an embodiment thereof will now be described, by way of example only, with reference to the accompanying drawings, in which:-

Figure 1 is a flow diagram illustrating an automated, computer-implemented procedure for characterisation of fraudulent transactions in accordance with the invention; and

Figure 2 is another flow diagram illustrating generation of a rule set in the Figure 1 procedure for use in characterisation of fraudulent transactions.

One example of an application of anomaly detection using the invention concerns characterisation of retail fraud committed in shops by cashiers. The invention in this example may be used in conjunction with current commercial systems that can measure and record the amount of money put into and taken out of cashiers' tills. Various kinds of cashier behaviour may indicate fraudulent or suspicious activity.

In this example of the invention transactions from a number of different cashiers' tills were employed. Each transaction was described by a number of attributes including cashier identity, date and time of transaction, transaction type (e.g. cash or non-cash) and an expected and an actual amount of cash in a till before and after a transaction. Each transaction is labelled with a single Boolean attribute which indicates "true" if the transaction is known or suspected to be fraudulent and "false" otherwise. Without access to retail fraud experts, definitions of background knowledge were generated in the form of concepts or functions relating to data attributes. One such function calculated a number of transactions handled by a specified cashier and having a discrepancy: here a discrepancy is a difference in value between actual and expected amounts of cash in the till before and after a single transaction.

In this example, the process of the invention derives rules from a training data set and the definitions of basic concepts or functions associated with data attributes previously mentioned. It evaluates the rules using a test data set and prunes them if necessary. The rules so derived may be sent to an expert for verification or loaded directly into a fraud management system for use in fraud detection. To detect fraud, the fraud management system reads data defining new events and transactions to determine whether they are described by the derived rules or not. When an event or transaction is described by a rule then an alert may be given or a report produced to explain why the event was flagged up as potentially fraudulent. The fraud management system will be specific to a fraud application.

Benefits of applying the invention to characterisation of telecommunications and retail fraud comprise:

- Characterisations in the form of rule sets may be learnt automatically (rather than manually as in the prior art) from training data and any available background knowledge or rules contributed by experts– this reduces costs and duration of the characterisation process;
- Rule sets which are generated by this process are human readable and are readily assessable by human experts prior to deployment within a fraud management system; and
- the process may employ relational data, which is common in particular applications of the invention – consequently facts and transactions which are

in different locations and which are associated can be linked together.

The process of the invention employs inductive logic programming software implemented in a logic programming language called Prolog. This process has an objective of creating a set of rules that characterises a particular concept, the set often being called a concept description. A target concept description in this example is a characterisation of fraudulent behaviour to enable prediction of whether an event or transaction is fraudulent or not. The set of rules should be applicable to a new, previously unseen and unlabelled transaction and be capable of indicating accurately whether it is fraudulent or not.

A concept is described by data which in this example is a database of events or transactions that have individual labels indicating whether they are fraudulent or non-fraudulent. A label is a Boolean value, 1 or 0, indicating whether a particular event or transaction is fraudulent (1) or not (0). Transactions labelled as fraudulent are referred to as positive examples of the target concept and those labelled as non-fraudulent are referred to as negative examples of the target concept.

In addition to receiving labelled event/transactional data, the inductive logic programming software may receive input of further information, i.e. concepts, facts of interest or functions that can be used to calculate values of interest e.g. facts about customers and their accounts and a function that can be used to calculate an average monthly bill of a given customer. As previously mentioned, this further information is known as background knowledge, and is normally obtained from an expert in the relevant type of fraud.

As a precursor to generating a rule set, Before learning takes place, the labelled event/transactional data is randomly distributed into two non-overlapping subsets – a training data set and a test data set. Here non-overlapping means no data item is common to both subsets. A characterisation or set of rules is generated using the training data set. The set of rules is then evaluated on the test data set by comparing the actual fraudulent or otherwise label of each event/transaction with the equivalent predicted for it by the inductive logic programming software. This gives a value for prediction accuracy – the percentage of correctly assessed transactions in the test data set. Testing on a different data set of hitherto unseen examples, i.e. a set other than the training data set, is a good indicator of the validity of the rule set.

The target concept description is a set of rules in which each rule covers or characterises a proportion of the positive (fraudulent) examples of data but none of the negative (non-fraudulent) examples. It is obtained by repeatedly generating individual rules. When a rule is generated, positive examples which it covers are removed from the training data set. The process then iterates by generating successive rules using unremoved positive examples, i.e those still remaining in the training data set. After each iteration, positive examples covered by the rule most recently generated are removed. The process continues until there are too few positive examples remaining to allow another rule to be generated. This is known as the sequential covering approach, and is published in Machine Learning, T. Mitchell, McGraw-Hill, 1997.

Referring to Figure 1, a computer-implemented process 10 involving applying the inductive logic programming software (referred to as an ILP engine) at 12 to characterising fraudulent transactions is as follows. Background knowledge 14 and a training data set 16 are input to a computer (not shown) for processing at 12 by the ILP engine running on the computer: this produces a set of rules 18. Rule set performance is evaluated at 20 using a test data set 22.

Referring now also to Figure 2, processing 12 to generate a set of rules is shown in more detail. Individual rules have a form as follows:

IF {set of conditions} THEN {behaviour is fraudulent} (1)

A computer search for each individual rule begins at 30 with a most general rule (a rule with no conditions): searching is iterative (as will be described later) and generates a succession of rules, each new rule search beginning at 30. The most general rule is:

IF { } THEN target_predicate is true (2)

This most general rule is satisfied by all examples, both positive and negative, because it means that all transactions and facts are fraudulent. It undergoes a process of refinement to make it more useful. There are two ways of producing a refinement to a rule as follows:

- addition of a new condition to the IF{ } part of the rule;
- unification of different variables to become constants or structured terms;

Addition of a new condition and unification of different variables are standard expressions for refinement operator types though their implementation may differ between systems. A condition typically corresponds to a test on some quantity of interest, and tests are often implemented using corresponding functions in the background knowledge . When a new condition is added to a rule, its variables are unified with those in the rest of the rule according to user-specified mode declarations. Unification of a variable X to a variable Y means that all occurrences of X in the rule will be replaced by Y. A mode declaration for a predicate specifies the type of each variable and its mode. A variable mode may be input, output, or a constant. Only variables of the same type can be unified. Abiding by mode rules reduces the number of refinements than may be derived from a single rule and thus reduces the space of possible concept descriptions and speeds up the learning process. There may be more than one way of unifying a number of variables in a rule, in which case there will be more than one refinement of the rule.

For example, a variable X may refer to a list of items. X could be unified to a constant value [ ] which represents an empty list or to [Y|Z] which represents a non-empty list with a first element variable Y and the rest of the list is represented by another variable Z. Instantiating X by such unification constrains its value. In the first case, X is a list with no elements and in the second case it must be a non-empty list. Unification acts to refine variables and rules that contain them.

Variables that are defined as being in constant mode must be instantiated by a constant value. Variables of constant type can further be defined by the user as either non-numerical or numerical constants.

If a constant is defined as non-numerical then a list of possible discrete values for the constant must also be specified by a user in advance. For each possible value of the constant, a new version of an associated refinement is created in which the value is substituted in place of the corresponding variable. New refinements are evaluated using an appropriate accuracy estimate and the refinement giving the best accuracy score is recorded as the refinement of the original rule.

If a constant is specified as numerical, it can be further defined as either an integer or a floating-point number. A method for calculating a best constant in accordance with the

invention applies to both integers and floating point numbers. If a constant is defined as numerical then a continuous range of possible constant values must be specified by a user in advance. For example, if the condition was "minutes_past_the_hour(X)" then X could have a range 0-59.

In an integer constant search, if a range or interval length for a particular constant is less than 50 in length, all integers (points) in the range are considered. For each of these integers, a new version of a respective associated refinement is created in which the relevant integer is substituted in place of a corresponding variable and new rules are evaluated and given an accuracy score using an appropriate accuracy estimation procedure. The constant(s) giving a best accuracy score is(are) recorded.

If the integer interval length is greater than 50, then the computer carries out a recursive process as follows:

1. A proportion of the points (which are evenly spaced) in the interval length are sampled to derive an initial set of constant values. For example, in the "minutes_past_the_hour(X)" example, 10, 20, 30, 40 and 50 minutes might be sampled. For each of these values, a new version of a respective refinement is created in which the value is substituted in place of a corresponding variable and a respective rule is evaluated for each value together with an associated accuracy estimate.

2. a. If a single constant value provides the best score then a number of the values (the number of which is a user selected parameter in the ILP engine 12) either side of this value are sampled. For instance, if the condition *minutes_past_the_hour(20)* gave the best accuracy then the following more precise conditions may then be evaluated:

   - *minutes_past_the_hour(15)*
   - *minutes_past_the_hour(16)*
   - *minutes_past_the_hour(17)*
   - *minutes_past_the_hour(18)*
   - *minutes_past_the_hour(19)*
   - *minutes_past_the_hour(21)*

- *minutes_past_the_hour(22)*
- *minutes_past_the_hour(23)*
- *minutes_past_the_hour(24)*
- *minutes_past_the_hour(25)*

If a single constant value in X = 15 to 25 gives the best accuracy score then that value is chosen as a final value of the constant X.

2. b. If more than one constant value provides the best score then if they are consecutive points in the sampling then the highest and lowest values are taken and the values in their surrounding intervals are tested. For example, if *minutes_past_the_hour(20)*, *minutes_past_the_hour(30)* and *minutes_past_the_hour(40)* all returned the same accuracy then the following points would be tested for accuracy :

- *minutes_past_the_hour(15)*
- *minutes_past_the_hour(16)*
- *minutes_past_the_hour(17)*
- *minutes_past_the_hour(18)*
- *minutes_past_the_hour(19)*
- *minutes_past_the_hour(41)*
- *minutes_past_the_hour(42)*
- *minutes_past_the_hour(43)*
- *minutes_past_the_hour(44)*
- *minutes_past_the_hour(45)*

If the accuracy score decreases at an integer value N in the range 15 to 19 or 41 to 45, then (N-1) is taken as the constant in the refinement of the relevant rule.

2. c. If a plurality of constant values provides the best accuracy score, and the values are not consecutive sampled points then they are arranged into respective subsets of consecutive points. The largest of these subsets is selected, and the procedure for a

list of consecutive points is followed as at 2b above: e.g. if *minutes_past_the_hour(20)*, *minutes_past_the_hour(30)* and *minutes_past_the_hour(50)* scored best then the subset *minutes_past_the_hour(20) – minutes_past_the_hour(30)* would be chosen. If the largest interval consists of only one value, then the procedure for a single returned value is followed as at 1. above.

The user can opt to conduct a beam constant search: here a beam is an expression describing generating a number of possible refinements to a rule and recording all of them to enable a choice to be made between them later when subsequent refinements have been generated. In this example, N refinements of a rule, each with a different constant value are recorded. This can be very effective, as the 'best' constant with highest accuracy at one point in the refinement process 32 may not turn out to be the 'best' value over a series of repeated refinement iterations. This avoids the process 32 getting fixed in local non-optimum maxima.

Some variables in conditions/rules may be associated with multiple constants: if so each constant associated with such a variable is treated as an individual constant, and a respective best value for each is found separately as described above. An individual constant value that obtains a highest accuracy score for the relevant rule is kept and the corresponding variable is instantiated to that value. The remaining variables of constant type are instantiated by following this process recursively until all constant type variables have been instantiated (i.e. substituted by values).

Once all refinements of a rule have been found, in accordance with the invention, the computer filters refinements at 34 to remove any rules that are duplicates or equivalents of others in the set. Two rules are equivalent in that they express the same concept if their conditions in the IF {set of conditions} part of the rule are the same but the conditions are ordered differently. For example, IF {set of conditions} consisting of two conditions A and B is equivalent to IF {set of conditions} with the same two conditions in a different order, i.e. B and A. One of the two equivalent rules is removed from the list of refinements and so is not considered further during rule refinement, which reduces the processing burden.

Additionally, in accordance with the invention, symmetric conditions are not allowed in any rule. For example, a condition equal(X,2) means a variable X is equal in value to 2,

is symmetric to equal(2,X), i.e. 2 is equal in value to a variable X. One of the two symmetric rules is removed from the list of refinements and so is not considered further.

Pruning refinements to remove equivalent rules and symmetric conditions results in fewer rules for the computer to consider at successive iterations of the refinement process 32, so the whole automated rule generation process is speeded up. Such pruning can reduce rule search space considerably, albeit the extent of this reduction depends on what application is envisaged for the invention and how many possible conditions are symmetric: in this connection where numerical variables are involved symmetric conditions are usually numerous due to the use of 'equals' conditions such as equal(Y,X). For example, in the retail fraud example, the rule search space can be cut by up to a third.

A 'less than or equals' condition referred to as 'lteq', and an 'equals' conditions are often used as part of the background knowledge 14. They are very useful conditions for comparing numerical variables within the data. For this reason, part of the filtering process 34 ascertains that equals and lteq conditions in rules meet checking requirements as follows:

- End of interval check: the computer checks the end of intervals where constant values are involved: e.g. a condition *lteq(A, 1000)* means variable A is less than or equal to 1000: it is unnecessary if *A* has a user-defined range of between 0 and 1000, so a refinement containing this condition is removed In addition, lteq(1000, A), 1000 is less than or equal to A, should be equals(A, 1000) as A cannot be more than 1000. Therefore, refinements containing such conditions are rejected.

- Multiple 'lteq' predicate check: if two conditions lteq(A,X) and lteq(B,X) where A and B are constants, are contained in the body of a rule, then one condition may be removed depending on the values of A and B. For example, if lteq(30,X) and lteq(40,X) both appear in a rule, then the computer removes the condition lteq(30,X) from the rule as being redundant, because if 40 is less then or equal to X then so also is 30.

- Equals and lteq duplication check: in accordance with the invention if the body of

a rule contains both conditions lteq(C, Constant) and equals(C, Constant), then only the equals condition is needed. Therefore, refinements containing lteq conditions with associated equals conditions of this nature are rejected by the computer.

Rule refinements are also filtered at 34 by the computer using a method called 'Encoding Length Restriction' disclosed by N. Lavrac and S. Dzeroski, Inductive Logic Programming: Techniques and Applications. Ellis Horwood, New York, 1994. It is based on a 'Minimum Description Length' principle disclosed by B. Pfahringer, Practical Uses of the Minimum Description Length Principle in Inductive Learning, PhD Thesis, Technical University of Vienna, 1995.

Where training examples are noisy (i.e. contain incorrect or missing values), it is desirable to ensure that rules generated using the invention does not overfit data by treating noise present in the data as requiring fitting. Rule sets that overfit training data may include some very specific rules that only cover a few training data samples. In noisy domains, it is likely that these few samples will be noisy: noisy data samples are unlikely to indicate transactions which are truly representative of fraud, and so rules should not be derived to cover them.

The Encoding Length Restriction avoids overfitting noisy data by generating a rule refinement only as long as the cost of encoding the refinement will not exceed the cost of encoding the positive examples covered by the refinement where 'cost' means number of bits. A refinement is rejected by the computer if this cost criterion is not met. This prevents rules becoming too specific, i.e. covering few but potentially noisy transactions.

Once a rule is refined, the resulting refinements are evaluated in order to identify those which are best. The computer evaluates rules at 36 by estimating their classification accuracy. This accuracy may be estimated using an expected classification accuracy estimate technique disclosed by N. Lavrac and S. Dzeroski, Inductive Logic Programming, Techniques and Applications. Ellis Horwood, New York, 1994, and by F. Zelezny and N. Lavrac, An Analysis of Heuristic Rule Evaluation Measures, J. Stefan Institute Technical Report, March 1999. Alternatively, it may be estimated using a weighted relative accuracy estimate disclosed by N. Lavrac, P. Flach and B. Zupan, Rule Evaluation Measures: A Unifying View, Proceedings of the 9th International Workshop

on Inductive Logic Programming (ILP-99), volume 1634 of Lecture Notes in Artificial Intelligence, pages 174-185, Springer-Verlag, June 1999. A user may decide which estimating technique is used to guide a rule search through a hypothesis space during rule generation.

Once refinements have been evaluated in terms of accuracy, they are then tested by the computer for what is referred to in the art of rule generation as 'significance'. In this example a significance testing method is used which is based on a likelihood ratio statistic disclosed in the N. Lavrac and S. Dzeroski reference above. A rule is defined as 'significant' if its likelihood ratio statistic value is greater than a predefined threshold set by the user.

If a rule covers n positive examples and m negative examples, an optimum outcome of refining the rule is that one of its refinements (an optimum refinement) will cover n positive examples and no negative examples. A likelihood ratio for this optimum refinement can be calculated by the computer. A rule is defined as 'possibly significant' if its optimum refinement is significant. Note that it is possible that a rule may not actually be significant, but it may be possibly significant in accordance with this definition.

The computer checks a rule under consideration in the process 12 at 38 to see whether or not it meets rule construction stopping criteria: in this connection, the construction of an individual rule terminates when the computer determines that any one or more of three stopping criteria is fulfilled as follows:

1. the number of conditions in any rule in a beam (as defined earlier) currently being processed is greater than or equal to a maximum rule length specified by the user. If a most significant rule (see at 2. below) exists this is added to the accumulating rule set at 40,

2. a most significant rule covers no negative examples – where the most significant rule is defined as a rule that is either present in the current beam, or was present in a previous beam, and this rule:

   a) is significant,

   b) obtained the highest likelihood ratio statistic value found so far, and

   c) obtained an accuracy value greater than the accuracy value of the most general rule (that covers all examples, both positive and negative), and

3. the previous refinement step 32 produced no refinements eligible to enter the new beam; if a most significant rule exists it is added to the accumulating rule set at 40.

Note that a most significant rule may not necessarily exist, if so no significant refinements have been found so far. If it is the case that a most significant rule does not exist but the stopping criteria at 38 are satisfied, then no rule is added to the rule set at 40 by the computer and the stopping criteria at 44 will be satisfied (as will be described later).

When a rule is added at 40, the positive examples it covers are removed from the training data by the computer at 42, and remaining or unremoved positive and negative examples form a modified training data set for a subsequent iteration (if any) of the rule search.

At 44 the computer checks to see whether or not the accumulating rule set satisfies stopping criteria. In this connection, accumulation of the rule set terminates at 46 (finalising the rule set) when either of the following criteria is fulfilled, that is to say when either:

- construction of a rule is terminated because a most significant rule does not exist, or
- too few positive examples remain for further rules to be significant.

If at 44 the accumulating rule set does not satisfy the rule set stopping criteria, the computer selects another most general rule at 30 and accumulation of the rule set iterates through stages 32 *etc.* At any given time in operation of the rule generation process 12, there are a number (zero or more) rules for which processing has terminated and which have been added in the accumulating rule set, and there are (one or more) evolving rules or proto-rules for which processing to yield refinements continues iteratively.

If evolving rules are checked at 38 and are found not to meet any of the rule construction stopping criteria previously mentioned, those refinements of such rules are chosen which have the best accuracy estimate scores. The chosen refinements then provide a basis for a next generation of rules to be refined further in subsequent refinement iterations.

The user defines the number of refinements forming a new beam to be taken by the computer to a further iteration by fixing a parameter called 'beam_width'. As has been said, a beam is a number of recorded possible refinements to a rule from which a choice will be made later, and beam_width is the number of refinements in it. For a beam width N, the refinements having the best N accuracy estimate scores are found and taken forward at 48 as part of the new beam to the next iteration. The sequence of stages 32 to 38 then iterates for this new beam via a loop 50.

Each refinement entering the new beam must:

- be possibly significant (but not necessarily significant), and
- improve upon or equal the accuracy of its parent rule (the rule from which it was derived by refinement previously).

If required by the user, the accumulated rule set can be post-pruned by the computer using a reduced error pruning method disclosed by J. Fürnkranz, A Comparison of Pruning Methods for Relational Concept Learning, Proceedings of AAAI'94 Workshop on Knowledge Discovery in Databases (KDD-94), Seattle, WA, 1994. In this case, another set of examples should be provided – a pruning set of examples.

Examples of a small training data set, background knowledge and a rule set generated therfrom will now be given. In practice there may be very large numbers of data samples in a data set

Training data

The training data is a transaction database, represented as Prolog facts in a format as follows:

trans(Trans ID, Date, Time, Cashier, Expected amount in till, Actual amount in till, Suspicious Flag). Here 'trans' and 'Trans' mean transaction and ID means identity. A sample of an example set of transaction data is shown below. Transactions with Suspicious Flag = 1 are fraudulent (positive examples), and with Suspicious Flag = 0 are not (negative examples).The individual Prolog facts were:

trans(1,30/08/2003,09:02,cashier_1,121.87,123.96, 0).
trans(2,30/08/2003,08:56,cashier_1,119.38,121.82, 0).
trans(3,30/08/2003,08:50,cashier_1,118.59,119.38, 0).
trans(4,30/08/2003,08:48,cashier_1,116.50,118.59, 0).

trans(5,30/08/2003,08:44,cashier_1,115.71,116.50, 0).
trans(6,30/08/2003,22:40,cashier_2,431.68,435.17, 0).
trans(7,30/08/2003,22:37,cashier_2,423.70,431.68, 1).
trans(8,30/08/2003,22:35,cashier_2,420.01,423.70, 0).

Background knowledge:
Examples of appropriate background knowledge concepts, represented using Prolog, are:

discrepancy(Trans_ID, Discrepancy).
This gives the discrepancy in UK £ and pence between the expected amount of cash in a till and the actual amount of cash in that till for a particular transaction, e.g.:

discrepancy(1, 2.09).
discrepancy(2, 2.44).
discrepancy(7, 7.98).

total_trans(Cashier, Total number of transactions, Month and Year).
This gives the total number of transactions made by the cashier in a given month and year, e.g.:

total_trans(cashier_1, 455, 08/2003).
total_trans(cashier_2, 345, 08/2003).

number_of_trans_with_discrepancy(Cashier, Number, Month and Year).
This gives the total number of transactions with a discrepancy made by a cashier in a given month and year, e.g.:

number_of_trans_with_discrepancy(cashier_1, 38, 08/2003).
number_of_trans_with_discrepancy(cashier_2, 93, 08/2003).

number_of_trans_with_discrepancy_greater_than(Cashier, Number, Bound, Month and Year).
This gives the total number of transaction with a discrepancy greater than some bound made by a cashier in a given month and year, e.g.:

number_of_trans_with_discrepancy_greater_than(cashier_1,5,100,08/2003).
number_of_trans_with_discrepancy_greater_than(cashier_1,3,150,08/2003).
number_of_trans_with_discrepancy_greater_than(cashier_2,15,100,08/2003)
number_of_trans_with_discrepancy_greater_than(cashier_2,2,200,08/2003).

discrepancy(Trans_ID, Discrepancy).

This gives the discrepancy between the expected amount of cash in the till and the actual amount of cash in the till for a particular transaction, e.g.:

discrepancy(1, 2.09).
discrepancy(2, 2.44).
discrepancy(7, 7.98).

total_trans(Cashier, Total number of transactions, Month and Year).

This gives the total number of transactions made by the cashier in a given month and year, e.g.:

total_trans(cashier_1, 455, 08/2003).
total_trans(cashier_2, 345, 08/2003).

number_of_trans_with_discrepancy(Cashier, Number, Month and Year).

This gives the total number of transactions with a discrepancy made by a cashier in a given month and year, e.g.:

number_of_trans_with_discrepancy(cashier_1, 38, 08/2003).
number_of_trans_with_discrepancy(cashier_2, 93, 08/2003).

number_of_trans_with_discrepancy_greater_than(Cashier, Number, Bound, Month and Year).

This gives the total number of transaction with a discrepancy greater than some bound made by a cashier in a given month and year, e.g.:

number_of_trans_with_discrepancy_greater_than(cashier_1,5,100,08/2003).
number_of_trans_with_discrepancy_greater_than(cashier_1,3,150,08/2003).
number_of_trans_with_discrepancy_greater_than(cashier_2,15,100,08/2003)
number_of_trans_with_discrepancy_greater_than(cashier_2,2,200,08/2003)

Generated rule set:

The target concept is fraudulent(Cashier). The rule set characterises a cashier who has made fraudulent transactions.

fraudulent(Cashier)
number_of_trans_with_discrepancy_greater_than(Cashier, Discrepancies, 100, Month),
Discrepancies ≥10.

fraudulent(Cashier) :-
total_trans(Cashier, Total_Trans, Month),

Total_Trans ≥ 455,
number_of_trans_with_discrepancy(Cashier, Discrepancies, Month),
Discrepancies ≥ 230.

This example of a generated rule set characterises fraudulent cashiers using two rules. The first rule indicates that a cashier is fraudulent if that in a single month, the cashier has performed at least 10 transactions with a discrepancy greater than 100.

The second rule describes a cashier as fraudulent if in a single month, the cashier has carried out at least 455 transactions, where at least 230 of these have had a discrepancy between the expected amount and the actual transaction amount

The embodiment of the invention described above provides the following benefits:

- It is fast because it prunes out duplicate rules avoiding unnecessary processing;
- It can deal with and tune numerical and non-numerical constants to derive rules that bound variables (e.g. IF transaction value is between £19.45 and £67.89 THEN ...);
- It can make use of many different heuristics (decision techniques e.g. based on scores for accuracy), which can be changed and turned on or off by a user;
- It uses a weighted relative accuracy measure in rule generation;
- It develops rules that are readable and its reasoning can be understood (unlike a neural network for example);
- It can be tuned to a particular application by adjusting its parameters and changing/adding heuristics;
- It can use relational and structural data that can be expressed in Prolog;
- It can process numerical and non-numerical data; and
- It can make use of expert knowledge encoded in Prolog.

The process undertaken by the ILP engine at 12 as set out in the foregoing description can clearly be evaluated by an appropriate computer program comprising program instructions embodied in an appropriate carrier medium and running on a conventional computer system. The computer program may be embodied in a memory, a floppy or compact or optical disc or other hardware recordal medium, or an electrical signal. Such a program is straightforward for a skilled programmer to implement on the basis of the

foregoing description without requiring invention, because it involves well known computational procedures.

Claims

1. A method of anomaly detection characterised in that it incorporates the steps of:-
   a) developing a rule set of at least one anomaly characterisation rule from a training data set and any available relevant background knowledge, a rule covering a proportion of positive anomaly examples of data in the training data set, and
   b) applying the rule set to test data for anomaly detection therein.

2. An automated method of anomaly detection characterised in that it comprises using computer apparatus to execute the steps of:-
   a) developing a rule set of at least one anomaly characterisation rule from a training data set and any available relevant background knowledge, a rule covering a proportion of positive anomaly examples of data in the training data set, and
   b) applying the rule set to test data for anomaly detection therein.

3. A method according to Claim 2 characterised in that data samples in the training data set have characters indicating whether or not they are associated with anomalies.

4. A method according to Claim 3 characterised in that it is a method of detecting telecommunications or retail fraud from anomalous data.

5. A method according to Claim 4 characterised in that it employs inductive logic programming to develop the rule set.

6. A method according to Claim 5 characterised in that each rule has a form that an anomaly is detected or otherwise by application of the rule according to whether or not a condition set of at least one condition associated with the rule is fulfilled.

7. A method according to Claim 6 characterised in that each rule is developed by refining a most general rule by at least one of:
   a) addition of a new condition to the condition set;
   b) unification of different variables to become constants or structured terms.

8. A method according to Claim 7 characterised in that a variable in a rule which is defined as being in constant mode and is numerical is at least partly evaluated by providing a range of values for the variable, estimating an accuracy for each value and selecting a value having optimum accuracy.

9. A method according to Claim 8 characterised in that the range of values is a first range with values which are relatively widely spaced, a single optimum accuracy value is obtained for the variable, and the method includes selecting a second and relatively narrowly spaced range of values in the optimum accuracy value's vicinity, estimating an accuracy for each value in the second range and selecting a value in the second range having optimum accuracy.

10. A method according to Claim 5 characterised in that it includes filtering to remove duplicates of rules and equivalents of rules, i.e. rules having like but differently ordered conditions compared to another rule, and rules which have conditions which are symmetric compared to those of another rule.

11. A method according to Claim 5 or 10 characterised in that it includes filtering to remove unnecessary 'less than or equal to' ("lteq") conditions.

12. A method according to Claim 11 characterised in that the unnecessary "lteq" conditions are associated with at least one of ends of intervals, multiple lteq predicates and equality condition and lteq duplication.

13. A method according to Claim 5 characterised in that it includes implementing an encoding length restriction to avoid overfitting noisy data by rejecting a rule refinement if the refinement encoding cost in number of bits exceeds a cost of encoding the positive examples covered by the refinement.

14. A method according to Claim 5 characterised in that it includes stopping construction of a rule if at least one of three stopping criteria is fulfilled as follows:

a) the number of conditions in any rule in a beam of rules being processed is greater than or equal to a prearranged maximum rule length,

b) no negative examples are covered by a most significant rule, which is a rule that:

i) is present in a beam currently being or having been processed,

ii) is significant,

iii) has obtained a highest likelihood ratio statistic value found so far, and

iv) has obtained an accuracy value greater than a most general rule accuracy value, and

c) no refinements were produced which were eligible to enter the beam currently being processed in a most recent refinement processing step (32).

15. A method according to Claim 14 characterised in that it includes adding the most significant rule to a list of derived rules and removing positive examples covered by the most significant rule from the training data set.

16. A method according to Claim 5 characterised in that it includes:

a) selecting rules which have not met rule construction stopping criteria,

b) selecting a subset of refinements of the selected rules associated with accuracy estimate scores higher than those of other refinements of the selected rules, and

c) iterating a rule refinement, filtering and evaluation procedure (32 to 38) to identify any refined rule usable to test data.

17. Computer apparatus for anomaly detection characterised in that it is programmed to execute the steps of:-

a) developing a rule set of at least one anomaly characterisation rule from a training data set and any available relevant background knowledge, a rule covering a proportion of positive anomaly examples of data in the training data set, and

b) applying the rule set to test data for anomaly detection therein.

18. Computer software for use in anomaly detection characterised in that it incorporates instructions for controlling computer apparatus to execute the steps of:-

a) developing a rule set of at least one anomaly characterisation rule from a training data set and any available relevant background knowledge, a rule covering a proportion of positive anomaly examples of data in the training data

set, and

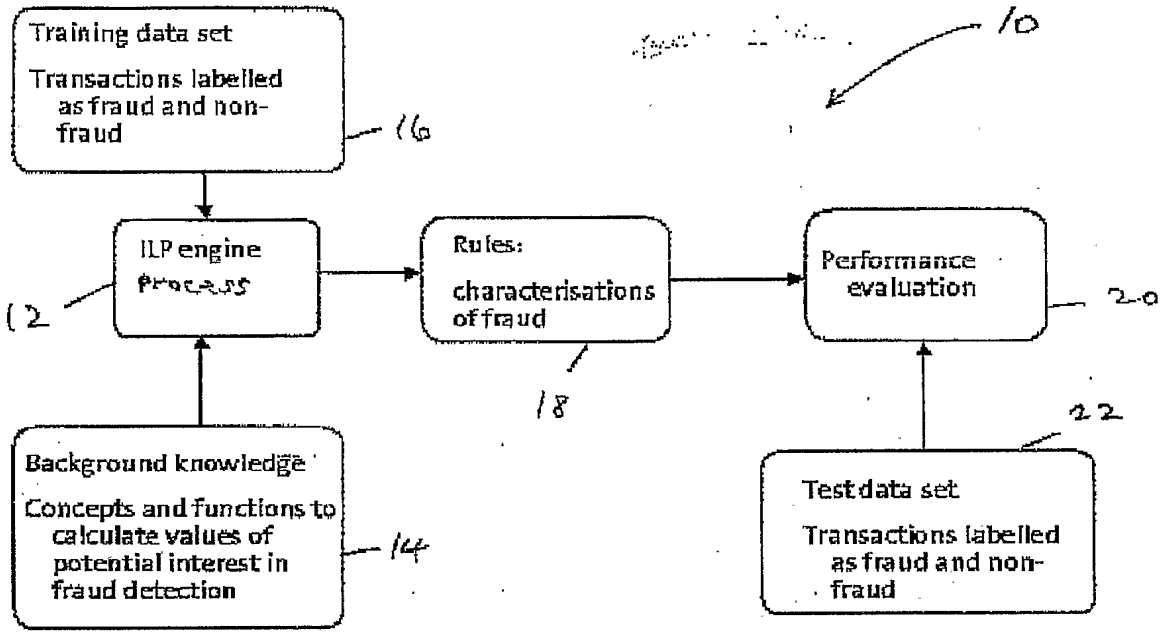
b) applying the rule set to test data for anomaly detection therein.

ABSTRACT

A method of anomaly detection applicable to telecommunications or retail fraud uses inductive logic programming to develop anomaly characterisation rules from relevant background knowledge and a training data set, which includes positive anomaly samples of data covered by rules. Data samples include 1 or 0 indicating association or otherwise with anomalies. An anomaly is detected by a rule having condition set which the anomaly fulfils. Rules are developed by addition of conditions and unification of variables, and are filtered to remove duplicates, equivalents, symmetric rules and unnecessary conditions. Overfitting of noisy data is avoid by an encoding cost criterion. Termination of rule construction involves criteria of rule length, absence of negative examples, rule significance and accuracy, and absence of recent refinement. Iteration of rule construction involves selecting rules with unterminated construction, selecting rule refinements associated with high accuracies, and iterating a rule refinement, filtering and evaluation procedure (32 to 38) to identify any refined rule usable to test data.

1/1

FIGURE 1



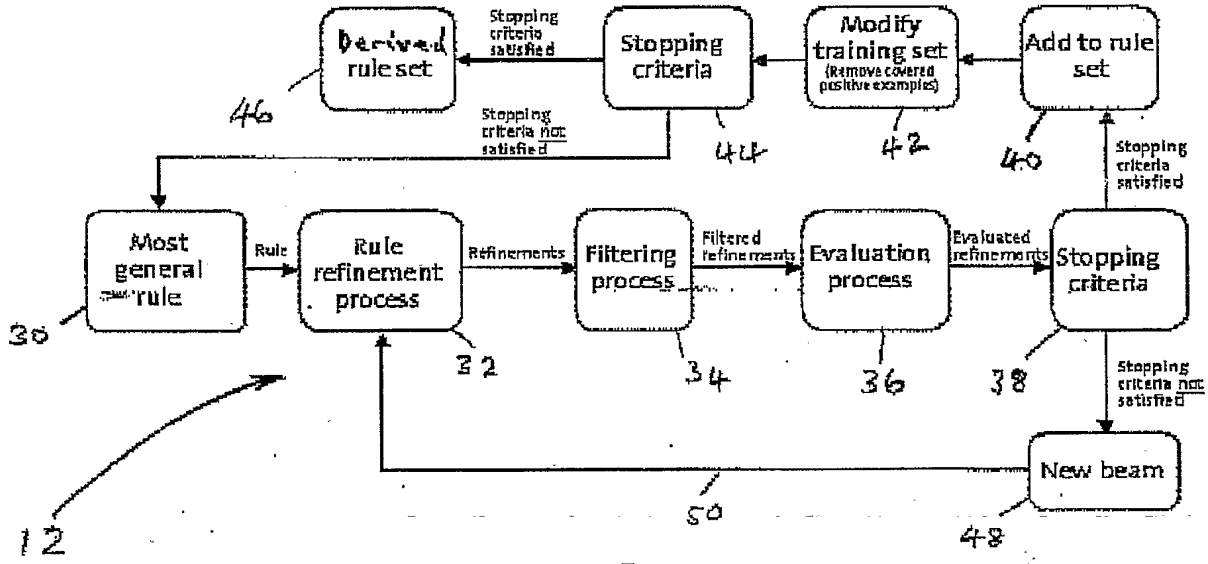


FIGURE 2